Research Article

ACCENTS Transactions on Information Security, Vol 2(8)
ISSN (Online): 2455-7196
http://dx.doi.org/10.19101/TIS.2017.28001

A secure key exchange protocol using link weights and dynamic tree parity machine (TPM)

Ganesan P[1*], Priyanka B.R[2], Mahfooz Sheikh[1], Murthy D.H.R[1] and G.K.Patra[1]
CSIR Fourth Paradigm Institute, Bangalore, Karnataka, India[1]
JNN College of Engineering, Shivamogga, Karnataka, India[2]

Abstract
Neural cryptography approaches probabilistic algorithm for exchange of keys with assured security. Artificial neural networks do not require complex number theory instead they work on basic arithmetic computation. Tree parity machine (TPM), one of the multi-layer neural network helps in generation of secret key between two ends by the process of synchronization. However the process can be mimicked by an attacker with the powerful majority flipping attack on TPM. The proposed technique mitigates Majority Flipping Attack (MFA) using link weights driven dynamic synchronization of TPM.

Keywords
Majority flipping attack, Link weights, Tree parity machine.

1.Introduction
The artificial intelligence concept has been inherited in cryptography for secure key exchange named as neural cryptography [1-4]. Tree Parity Machine which is the main building blocks of neural key exchange is developed and used for secure communication. In this approach, the participants only need to perform basic arithmetic operations instead of number theory approach [5-8]. The two participants who share their output and input bits in public domain can securely synchronize each other by tree parity machine architecture. This architecture can be programmed into a hardware called field programmable gate array (FPGA). This is a lucrative model to implement in hardware because it needs to perform simple calculation for key exchange and memory constraints are less compared to number theoretic approach. This class of light-weight cryptography model should not compromise in security with the effect of geometric attacks [9]. The proposed work mainly focuses on overcoming geometric attack and majority flipping attack. The TPM, feed forward neural network consists of artificial neurons and structured as shown in the *Figure 1*. The network has three layers: input layer, hidden layer and output layer.

Input layer takes binary input of form {-1 or +1} and an initial random weight vector is assigned to the links connecting input neuron and the hidden neuron [2]. Hidden neurons output is calculated using sign function as per the equation (1). The hidden layer neurons are connected to an output neuron and output bit is computed using equation (2). The input vector $X_{i,j}$ and output bit of TPM network $\tau^{A/B}$ are public parameters where the weight vector $W_{i,j}$ and hidden neurons output $\sigma_j^{A/B}$ are private.

$$\sigma_j = \text{sign}\left(\sum_{j=1}^{N} W_{ij} * X_{ij}\right) \quad (1)$$

$$\tau = \prod_{i=1}^{K} \sigma_i \quad (2)$$

Two participants using the TPM network share the output bit and input vector on public medium synchronizes to a time dependent weight vector [4]. Mutual learning is a concept behind the synchronization of TPM [3].

If output of two TPMs A and B are equal, the network trains the weights of the hidden neuron which is having similar value as that of the output of network. The perceptrons have been trained using any one of the learning rules given in the equation (3) (4) (5). This stochastic process of learning forces attractive and repulsive behaviour on the Tree Parity Machine. When the attractive forces lead the repulsive forces, the two TPMs converge to identical weights which are translated to symmetric keys for encryption and decryption [1].

*Author for correspondence

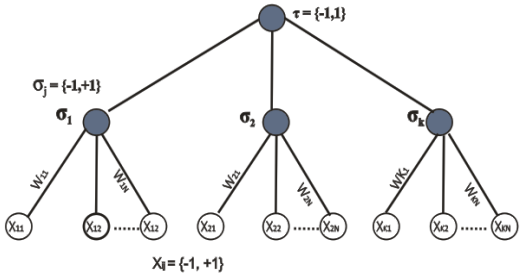

Figure 1 Structure of tree parity machine

Hebbian Rule:

$$W_{ij}^{+} = W_{ij} + X_{ij} * \tau \quad (3)$$

Anti-Hebbian Rule:

$$W_{ij}^{+} = W_{ij} - X_{ij} * \tau \quad (4)$$

Random Walk:

$$W_{ij}^{+} = W_{ij} + X_{ij} \quad (5)$$

2.Dynamic synchronization

A. Dynamic perceptron's

In order to improve the security of key exchange, a tricky synchronization has been applied on TPM. The number of hidden units is selected randomly at every time step, i.e., K is selected among four values 3, 4, 5, and 6 and the input vector size is determined by total number of weights divided by K. The K value will be shared in public domain but it will also be kept as private at constant time steps. K is calculated by approximating the sigma of link weights at constant iterations also picking it as an individual random value at different constant iterations [5].

A counter is incremented if τ^A and τ^B are equal and counter is reset to zero whenever the values are not equal. The counter is used to perceive the amount of synchronization on a network [6]**.** Random selection of K is dropped while counter increases beyond the threshold for the network to synchronize eventually. If K is random, TPM will experience repulsive forces while learning the network. If K is approximated by link weights, TPM would most probably experience attractive forces while learning the network. This alternative force induces security during the synchronization.

K1= K2= K3= rand{3,4,5,6} (6)

K1, K2 and K3 are hidden neurons of TPM A, TPM B and TPM E respectively.

B. Link weights

Synaptic depth of weights signifies security of TPM. Increasing the value of synaptic depth decreases the probability of success for attacker. Synaptic depth can be increased by attaching a link weight to the TPM weight. Link weights are also trained using new learning methods. Whenever a weight hits the boundary i.e. –L or L, the corresponding link weight undergoes learning. If the weight exceeds the boundary, the excess is transmitted to link weights. Interestingly the link weights of Tree Parity Machines synchronize to identical time dependent weights. Lower the synaptic depth of link weights, faster the convergence of link weights.

The dynamic TPM with link weights is shown in the *Figure 2***.** As the link weight synchronizes, the hidden layer output of link weight is determined considering X_i in input layer and can be expressed as shown in equation (7),

$$\sigma_j^{LA/LB} = f\left(\sum_{i=1}^{N} X_i * LW_i\right) \quad (7)$$

f can be sign function, threshold function or any other function specific to the problem under consideration [4]. Here output of f would be 0 and 1 instead of -1 and 1 in order to compute $D^{A/B}$ and $\sigma_j^{LA/LB}$ can be used to create a new code using equation (8) and (9) which would be approximately equal between the synchronizing networks. This code has been used to evaluate K value of TPM. K can be inputted to TPM and it will not be shared in public domain. There is a high probability that two machines would have equal K when its link weight vector synchronizes to identical time dependent weights. Dynamic TPM uses this knowledge to predict K value at particular constant time steps.

$$D^{A/B} = \sum_{j=0}^{K-1} \left(2^j \sigma_j^{LA/LB}\right) \quad (8)$$

$$K^{A/B} = \left(D^{A/B} \bmod 4\right) + 3 \quad (9)$$

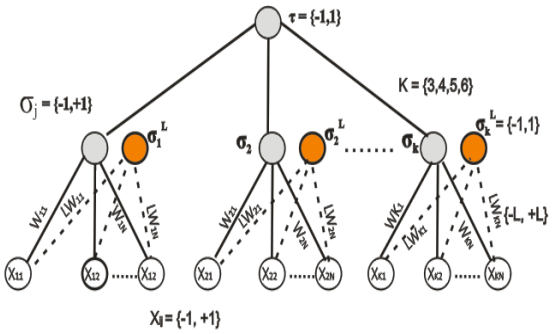

Figure 2 TPM with link weights

C. Intel digital random number generator

The IVY bridge processor is manufactured with digital random number generator (DRNG). National

institute of standards and technology (NIST) recommends Intel DRNG for creating cryptographic applications. This random number is compliant with security and other cryptographic standards such as NIST SP800-90, FIPS-140-2 and ANSI X9.82 [8]. Intel DRNG has RDRAND and RDSEED instructions to fetch random number from processor [7]. RDRAND instruction is used in TPM implementation to generate digital random number for initial random weights and momentary inputs.

3.Security analysis

An Eavesdropper, who listen this network having similar architecture and accessing every public parameter, has disadvantages in synchronizing to genuine participant. The difficulties are, being synchronized to the link weights network [10]. As the link weight grows whenever its corresponding weight hits the limit L, the attacker link weight may not be same at particular instant of time. This is because of attacker's weight (W_i^E) is not equal to TPM weight ($W_i^{A/B}$) at that instant of synchronization. Probability of success for attacker is high when synchronizes to legitimate TPMs which is depicted in *Figure 3*. *Figure 4* shows number of iterations taken for TPM to synchronize with other network. Probability of attacker's success is expressed by the equation (10)**.** Probability of success decreases exponentially with the increase of L [1]. *Figure 5* shows that success probability of attacker is reduced exponentially with advent of proposed technique. Since the link weight ($LW_i^{A/B}$) learns whenever its related weight ($W_i^{A/B}$) exceeds synaptic depth L1, the dynamic TPM is indirectly increasing the value L1. Thus the probability of success for attacker is exponentially reducing. *Figure 6* depicts the number of iterations taken for dynamic TPM to synchronize which is linearly increased than TPM synchronization.

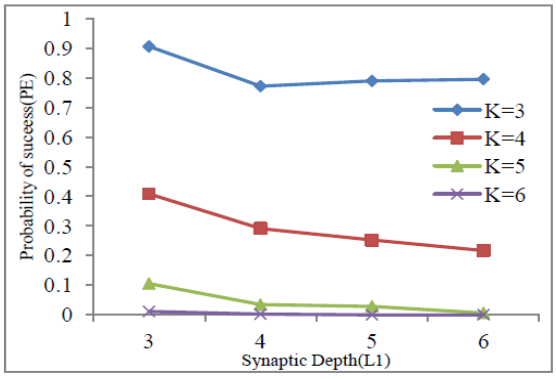

Figure 3 Probability of attacker's success against synaptic depth L in TPM average of 1000 runs

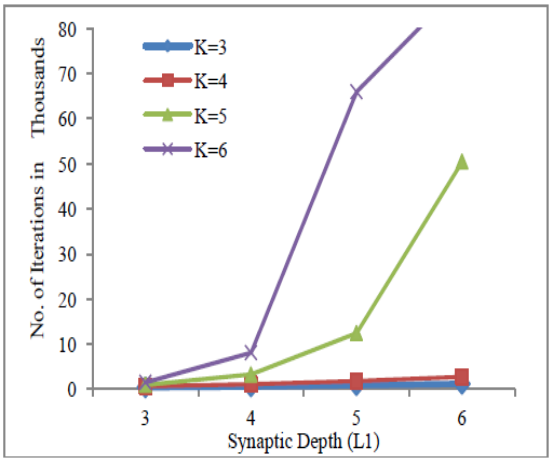

Figure 4 Number of iterations taken against synaptic depth L in TPM averaged out of 1000 runs

$$P_e \infty e^{-yL} \tag{10}$$

The empirical results shows that number of link weights ($LW_i^{A/B}$) synchronized is greater than or equal to number of weights ($W_i^{A/B}$) synchronized after 2/3 of total iterations in Dynamic TPM. Attacker's weights (W_i^E) synchronized are lesser than weights (W_i^E) synchronized in DTPM. Therefore there is a difference between link weights ($LW_i^{A/B}$) synchronized in TPM and link weights (LW_i^E) synchronized in attacker's machine. On the other hand, synchronized is closer to synchronized in DTPM. This difference is quantified and made useful for the legitimate participants to code K value [11].

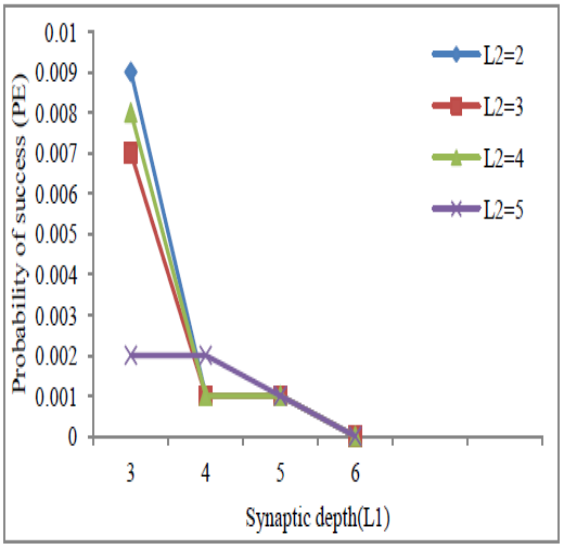

Figure 5 Probability of attacker's success against synaptic depth L in Dynamic TPM average of 1000 runs

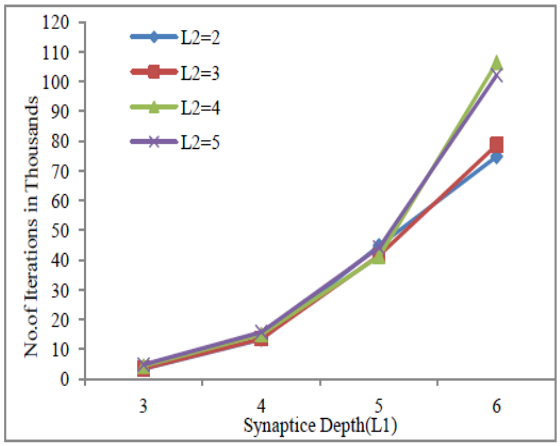

Figure 6 Number of iterations taken against synaptic depth l in dynamic TPM average of 1000 runs

4.Conclusion

The problem of secure communication is extensively analyzed using TPM by introducing dynamic evaluation and prediction of number of hidden neuron K. The problem has been solved in the proposed model using link weights driven dynamic synchronization of TPM and the results are significantly improved the security of cryptographic key exchange.

Acknowledgment

This work is developed as part of the CySeRO component of ARiEES project at CSIR-4PI, Bengaluru. Authors are thankful to the Head, CSIR-4PI for the support provided to carry out this project.

Conflicts of interest

The authors have no conflicts of interest to declare.

References

[1] Ruttor A, Kinzel W, Shacham L, Kanter I. Neural cryptography with feedback. Physical Review E. 2004; 69(4):046110.

[2] Menezes AJ, Van Oorschot PC, Vanstone SA. Handbook of applied cryptography. CRC press; 1996.

[3] Santhanalakshmi S, Sudarshan TS, Patra GK. Neural synchronization by mutual learning using genetic approach for secure key generation. In international conference on security in computer networks and distributed systems 2012 (pp. 422-31). Springer Berlin Heidelberg.

[4] Revankar P, Gandhare WZ, Rathod D. Private inputs to tree parity machine. International Journal of Computer Theory and Engineering. 2010; 2(4):665-9.

[5] Volkmer M, Wallner S. Tree parity machine rekeying architectures. IEEE Transactions on Computers. 2005; 54(4):421-7.

[6] Prabakaran N, Vivekanandan P. A new security on neural cryptography with queries. International Journal of Advanced Networking and Applications. 2010; 2(1):437-44.

[7] Hofemeier G. Intel digital random number generator (DRNG) software implementation guide.2012.

[8] Barker EB, Kelsey JM. Recommendation for random number generation using deterministic random bit generators (revised). US department of commerce, technology administration, National institute of standards and technology, computer security division, information technology laboratory; 2007.

[9] Ruttor A. Neural synchronization and cryptography. arXiv preprint arXiv:0711.2411. 2007.

[10] Kanter I, Kinzel W, Kanter E. Secure exchange of information by synchronization of neural networks. EPL (Europhysics Letters). 2002; 57(1):141-51.

[11] Klimov A, Mityagin A, Shamir A. Analysis of neural cryptography. In international conference on the theory and application of cryptology and information security 2002 (pp.288-98). Springer Berlin Heidelberg.

This paper is selected from proceedings of National Workshop on Cryptology-NWC 2016 organized at JNN College of Engineering Shimoga, Karnataka, India during 11-13, August 2016.